A Security Analysis of OpenID

Bart van Delft[1] and Martijn Oostdijk[2]

[1] Comp.Sci.Dept., Radboud Univ., P.O.Box 9010, 6500 GL, Nijmegen, The Netherlands, b.vandelft@student.ru.nl
[2] Novay, P.O.Box 589, 7500 AN, Enschede, The Netherlands, martijn.oostdijk@novay.nl

Abstract. OpenID, a standard for Web single sign-on, has been gaining popularity both with Identity Providers, Relying Parties, and users. This paper collects the security issues in OpenID found by others, occasionally extended by the authors, and presents them in a uniform way. It attempts to combine the shattered knowledge into a clear overview. The aim of this paper is to raise awareness about security issues surrounding OpenID and similar standards and help shape opinions on what (not) to expect from OpenID when deployed in a not-so-friendly context.

1 Introduction

In 2005 Brad Fitzpatrick developed the first version of the OpenID protocol. Initially intended to save users the effort of going through a registration process on multiple social (blogging) websites they wanted to join, the protocol has by now become a very popular single sign-on method on a large number of high profile websites (JanRain claims over 30,000 OpenID enabled websites in January 2009 [5]). In fact, given an average user on the Internet, chances are high that he or she already owns a so-called 'OpenID enabled login', since Google, Yahoo, Facebook, AOL and Windows Live accounts all support OpenID.

The latest version of the OpenID Authentication specification, version 2.0 [20], was released in December 2007[3]. The new version mainly enabled more flexibility and support for the possible identities to use, improved some security aspects and enabled the usage of extensions for attribute exchange.

This paper studies the security aspects of the OpenID 2.0 specification and its implementations. Information in this area has been around for some time. During the BlackHat USA conference in 2007, Eugene and Vlad Tsyrklevich listed a number of potential issues [22], and a similar collection can be found on [15]. Several singular findings have been reported on blogs and wikis as well, but this information is scattered and the possible impact is difficult to assess. By collecting these issues and looking at the underlying mechanisms of reported attacks, this paper contributes to the existing literature by trying to pinpoint the root causes and possible solutions of OpenID security issues.

[3] The original version written by Brad Fitzpatrick had no version number but is commonly referred to as OpenID 1.0. OpenID 2.0 is based on OpenID 1.1, which is a revised version of the original version from Fitzpatrick.

In relation to this paper, the original goal of OpenID should be kept in mind. OpenID was meant for authentication to 'simple' websites, blogs and the like. It was not intended for high-trust applications, such as e-banking. The requirement to ensure that claims about a user are actually correct, the so-called *level of assurance*, was low.

The OpenID protocol is not unique in its goals, nor in its methods. Several other web single sign-on specifications have been drafted with roughly the same targets in mind, the *SAML Web Browser Single Sign On* profile probably being the most notable [7]. The main differences between SAML and OpenID is that SAML requires the web applications involved to know each other beforehand, and that it is part of a far larger specification. The SAML specification was drafted by industrial companies whereas OpenID is a community-driven project. Other specifications in (roughly) the same category include OAuth [17] and WS-Trust [11], and Information Card [16].

1.1 OpenID from the user's perspective

To explain the OpenID concepts this section presents an example scenario from the user's perspective. The example introduces terminology used throughout the remainder of this paper.

A *User* is going to register himself at the web application `springnote.com` (the *Relying Party, RP*). Instead of filling out yet another registration form he provides the RP a URL that represents his identity, say `www.johndoe.com` (the *Identity URL* or *claimed identifier*). At this URL the RP can *discover* how it should verify the user's ownership of this URL. In this example the RP discovers that the owner of this URL should be able to log on at `openid.yahoo.com` (the *OpenID Provider, OP*) with username `jdoe`.

The RP redirects the user to the OP where he is asked to enter the password for username `jdoe`. After authentication of this *local identity* the user is asked whether he indeed wants to use his identity on `*.springnote.com` (the *realm*) and send along the *attributes* as requested by the RP (such as the user's full name and email address). If confirmed the user is redirected back to the RP.

Using a direct connection the RP ensures that the local identity is indeed authenticated by the OP. Assuming that no one except the owner can alter the content of `www.johndoe.com`, `springnote` is assured that this user is in fact the owner of this URL. The RP can now create a local account on its web application for this identity or, in the case of a returning user, log him in using this identity.

1.2 OpenID from a technical perspective

In order to understand all of the security issues described in Section 2, some knowledge of the OpenID 2.0 specification [19] is required. This section gives a global introduction to the protocol. Figure 1 gives a schematic overview of the protocol.

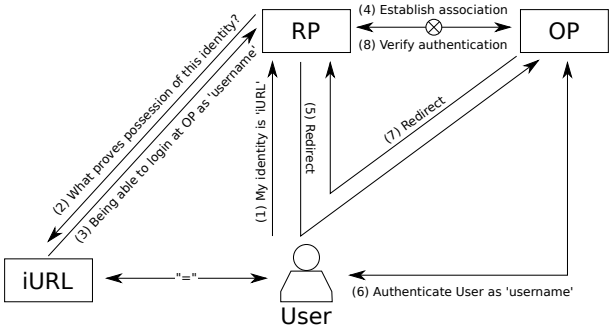

Fig. 1. Schematic overview of the protocol. Note that there is an exclusive OR-relation between step (4) and step (8).

Discovering the user's identity The RP fetches the Identity URL as specified by the user in order to discover his OP. In the current version there can be either a static link stating the OP and the local identity, or an XRDS-document can be used enabling the possibility of specifying different OPs for multiple protocols. In the latter case Yadis[4] is used to discover the OP and protocol version to use.

Establishing an association An *association* is used to ensure that no one but the OP has authenticated the user's local identity. Instead of using associations the RP could also choose to skip this step and validate an identity assertion at the end of the protocol run.

An association is established between the RP and the OP and consists of an *association_handle*, which is a unique name for this association, and a *MAC key* (Message Authentication Code) for signing messages. There are two methods to establish an association; one using Diffie-Hellman (DH) key exchange [4], and one requiring a secure channel (e.g. SSL).

Redirecting to the OP After the discovery of the OP and the optional set-up of an association, the RP forwards the user via a web redirect to the OP. This request again contains a number of parameters, most notable the claimed identity, the association handle (if applicable), the domain of the RP that needs to be allowed (the *realm*) and a URL at the RP where the user agent should be redirected after authentication (`return_to`).

Redirecting to the RP When the OP has authenticated the user's local identity and the user has allowed his identity to be used on the specified realm, the OP redirects the user to the RP with a number of parameters. These include the

[4] The Yadis protocol was developed by (roughly) the same community as OpenID, but is in its functionality not linked to it. Yadis used to be the original name for OpenID, being an acronym for "Yet Another Distributed Identity System".

association handle (if applicable), a server-time based nonce to prevent replay attacks and a signature over the values of a number of these parameters, including the parameters mentioned here. If an association was established this signature is based on the MAC key and signing algorithm specified in that association.

Verifying the response If no association had been established the RP still has to ensure the response was indeed generated by the OP. The RP sends the parameters received directly to the OP and the OP responds with, amongst others, an `is_valid` parameter stating whether the signature was his.

2 A classification of security issues

This section categorises security issues found into four categories. The categories represent different features of the goal that OpenID is trying to achieve, of typical implementation practice, of the OpenID specification, and of the underlying infrastructure of the Web.

Similar collections and discussions can be found in [8–10, 12, 14, 22].

Single Sign On The goal that OpenID is trying to achieve is usually referred to as Single Sign On (SSO) for the Web. The central idea behind SSO is that once a user has authenticated to an OP, he or she does not have to do so again. Within the context of typical OpenID provider implementations this also means that once a user has granted a permanent approval for a RP, this request for approval is not shown again.

Open to anyone An important feature of OpenID is that a user is able to choose his or her own Identity (URL), OP, and RPs. In principle, anyone can join and implement an OP or start accepting OpenID transactions as a RP.

OpenID-specific issues Some specific choices made in the OpenID specification, or lack thereof, lead to differences and possible weaknesses in implementations.

Use of Web standards Web-redirects are used to send the user from the RP to the OP and back again. While this makes light-weight requirements for OP, RP, and user, it also means that OpenID inherits weaknesses present in this redirection mechanism.

This distinction makes clear that not all vulnerabilities found threaten OpenID alone. Some are also applicable to other standards. The issues corresponding to these categories are listed in the sections below. Some of the issues fall into multiple categories.

2.1 The Single Sign On concept

Adversaries will focus on OPs A user uses the same OP to log in to multiple RPs. If an attacker is able to steal these credentials he obtains access to multiple

RPs including the data the user has stored there. The credentials stored at an OP become more valuable with the popularity of the OpenID standard.

On the other hand, once OpenID becomes the de facto standard for Web SSO, OPs are able to focus solely on the protection of its users data and implementation can be hardened.

When registering on a non-OpenID enabled website, a valid email address is often required. If a user forgets his password, this password or a reset option can be sent to his email address. Thus if an attacker successfully hacks a large email provider, e.g. Live mail, his gain will be roughly the same. A main difference is that in the OpenID scenario an attacker will not need to change the user's password, making the actions harder to detect.

Spying OPs As a result of always using the same OP to log in, the OP has the possibility to log every transaction with every RP the user visits. It knows the frequency, what times of the day etc. [22]. The OP's effort is just the logging of every transaction. Data mining tools are widely available for handling the resulting data, making this a feasible attack.

Compared with an OpenID-less setting, only the user's web browser is in a comparable position. Consider the Google Chrome web browser that sends every URL a user visits to the Google datacenters where these are (anonymously?) processed for improvement of the browser and other Google products. The concept of a (service-)provider you use for visiting other website collecting data on you is thus not new.

Cross-site scripting When a user is logged in to his OP and has permanently granted permission for a RP, the user will not be prompted with either the OP's login screen nor with the request for identifying himself to this specific RP. This allows for an extension of existing cross-site scripting attacks. An attacker could place a hidden frame on a website loading the OpenID-login from the targeted RP, in that manner logging the visitor in to that RP without this victim being aware. The attacker can now perform cross-site request forgery (XSRF) attacks such as performing actions on the RP in that user's name [22].

As a result, "logging out when finished" as is commonly advised, no longer guarantees protection against XSRF attacks. XSRF protection needs to be in place at the RP, as will be described in Section 3.3.

A different attack in this category is *clickjacking* [6] where stylesheet features are abused to make a user think he clicks a link (on a website controlled by the attacker) while in fact clicking a translucent button or link above it. In this manner some interaction can be accounted for.

Session Swapping In this attack an adversary tricks the user into visiting a replica page the user assumes to be a certain RP. The website however abuses the no-interaction log on to automatically log the user on to the real RP under an account controlled by the attacker. Being accustomed to the SSO-principle,

the user will be assuming he was signed on to his own account [10]. In order to reduce the risk of the user noticing the session swap the attacker should be aware of personal information of the user on the RP, such as username, personal background image etcetera.

If successful, the potential gain of an attacker would be search phrases or credit card information entered by the user on the RP in the assumption that this is his own account. Being the real owner of the account, the attacker can later retrieve this information by logging on to the RP. A large amount of luck is however required.

2.2 Open to anyone

The port-scanning RP At the beginning of the protocol the RP fetches the Identifier URL as specified by the user. Since the user is free in choosing the value of this URL, an adversary could enter URLs such as `http://www.example.com:21`, hereby abusing the RP as a (proxied) port-scanner. It might, depending on the RP's configuration, even be possible to scan internal locations such as `192.168.1.45:3654` [22].

Port-scanning in itself is not much of a threat, but having the possibility to scan addresses *within* the local network of the RP and the fact that this scanning easily can be automated makes it a possibly useful component in a targeted attack.

RP getting lost in discovery This attack abuses the fact that the RP has to connect to the Identifier URL as provided by the user. A malicious user could use large or endless files as Identifier URLs in an attempt to perform a DoS (Denial of Service attack) on the RP [22]. He does however first have to find an RP vulnerable to this attack, i.e. an RP that does not halt after hitting a limit during the fetching of a URL. Since it is trivial to mount such an attack, it is highly likely someone will try.

2.3 OpenID-specific issues

Diffie-Hellman / Man in the Middle attack As described in Section 1.2, Diffie-Hellman key exchange may be used to establish an association and send the corresponding MAC key from the OP to the RP. DH key exchange is vulnerable to a *Man in the Middle* (MitM) attack, where an attacker is acting as the OP to the RP and as the RP to the OP. This results in the attacker being able to change all the data being send from the OP to the RP, without the RP noticing that the signature is incorrect. Depending on the location of this Man in the Middle, he could either log in to one RP as being any user, or log in as any user from one OP to any RP [22].

In order to perform a MitM attack, the attacker either needs to have something in between the two parties (the RP and the OP) such as a router functioning as a proxy, or perform a DNS spoofing attack.

As long as RPs using OpenID only require low-level security (e.g. blogs, comment-systems) the effort of performing a MitM will probably outrank the gain. However, when large web shops or online payment companies start accepting OpenID intensively, a MitM attack becomes more tempting for an attacker.

Association poisoning In this attack an attacker sniffs the name of an association (the association handle) for a specific RP / OP communication. He then logs in to the RP using his own OP. Since the OP gets to choose the handle, the attacker chooses the exact same handle he just sniffed, making a vulnerable RP *overwrite* the original MAC key with the one specified by the attacker. The attacker may now, for a limited amount of time (if or until the real OP overwrites the handle again), modify the attributes sent from the OP to the RP, using his own signature to sign. If the attacker does not change these messages, the RP will think that the message from the OP are incorrectly signed and thus rejected [2].

To alter the data sent from the OP to the RP, the attacker has (as in the MitM attack) to install a node in between the two parties. Otherwise the only result will be that the messages from the original OP will be rejected by the RP. A quick review on *DotNetOpenId* and *JanRain's OpenID Ruby library* by [2] and a review on *openid-php* by the authors show that this attack cannot be performed against these commonly used libraries.

OpenID recycling A user may change from OP over time, leaving his local identity to be taken by a new user. If his OP is at the same time the supplier of his Identity URL (which is the case if e.g. username.myid.net or the Google / Yahoo! etc. buttons are directly used) this means that not only his local identity, but also his Identity URL are now owned by the new user. Without precautions the new user could log in to an RP and find the data of the previous owner of that Identity URL, since the RP cannot differentiate between them. The new owner does not have to be an adversary for this situation to arise.

The OpenID 2.0 Specification does provide a recommended solution to this problem in the form of *fragments* (section 11.15.1 in the specification). However because OPs are not *required* to adopt this solution, identity recycling issues might still occur.

2.4 Use of Web standards

Phishing In a phishing attack a user is tempted to visit a website impersonating the login-screen of the user's OP. If he fails to notice that this is not the official website of the OP and enters his credentials, the attacker is able to copy these credentials [14] [21].

Previous phishing attacks often required the user to click a suspiciously formatted link, but the addition of OpenID's redirection setup makes it possible to create a less questionable attack. An RP controlled by an adversary could simply redirect users to a phishing OP instead of to their real OP, without the

users noticing anything different from a regular login. The risk of being phished therefore increases with the introduction of OpenID.

Realm spoofing This attack [14] assumes the existence of a RP having an XSS- or other vulnerability enabling forwarding to other websites, e.g. `http://www.example.com?goto=http://evil.com`. An attacker creates a website imitating this RP and if the user tries to log in using OpenID the attacker tells the OP that his realm is the one of the official RP and in the return-to URL (Section 1.2) abuses the forward possibility that will redirect the user back to his website.

If successful, both the OP and the user will believe that the attributes are send to the official RP, but these are, although the OP compared the realm and the return-to URL correctly, in fact redirected to the imitating RP of the attacker.

To summarise the results of this section, Table 1 provides an overview of the security issues. The rightmost two columns of the table gives qualitative risk estimates for two scenarios: one in which OpenID is used for services with level of assurance (LOA, see [18]) equal to 1, and one in which OpenID is used for services with a higher LOA. The risk estimates are based on the combination of effort required for the specific attack, and the gain an attacker may receive from a successful attempt. The main difference between the two scenarios is based on the assumption that services with a higher LOA imply more potential gain for the attacker, increasing the likeliness that a specific vulnerability will be abused.

As may be derived from this table, the average risk in a LOA of 1 is roughly moderate, while the average risk for services with a higher LOA would be high when OpenID is deployed in that scenario.

			LOA	
Category	**Vulnerability**	**Threat**	**= 1**	**> 1**
Single Sign On	Always use same OPs	Attacks focus on them	L	M
		OP tracks its users	M	M
	No-interaction log in	Cross-site scripting	H	H
		Session Swapping	L	M
Being Open	Specify any Identity URL	Port-scanning RP	M	M
		RP lost in discovery	H	H
OpenID-specific	Usage of Diffie-Hellman	MitM attack	L	H
	Mild specification	Association poisoning	M	H
		OpenID recycling	-	-
Use of Web standards	RP redirects user to OP	Phishing	H	H
	User is in a browser	Clickjacking	M	H
		Realm spoofing	-	-

Table 1. An overview of the security issues listed in this paper. L = Low, M = Moderate, H = High risk.

3 An overview of solutions

This section lists possible directions for solutions to the issues described in Section 2. The solutions are grouped into a number of categories to show how they may be applied. Figure 2 gives an overview.

3.1 Trust frameworks

Most of the problems arising from the 'Open' part of OpenID can be solved by becoming more closed, e.g. via the usage of a trust framework between Identities, OPs and RPs. If these parties get whitelisted all attacks requiring the attacker to control one or more of these parties can be prevented. As a consequence it will no longer be possible to add an OpenID-login to every potential RP, neither will the end-users be able to use any Identifier or OP of their liking.

As an example the Federal Identity, Credential and Access Management (ICAM) committee of America recently profiled OpenID for Level of Assurance 1 [18] transactions with the Federal government [13]. In this profile a whitelist at the ICAM website specifies the OPs that RPs can trust during the authentication process.

3.2 Anti-phishing techniques

There are two main directions in preventing phishing attempts. First the OP can add personalisation elements such as a personal icon to its login-pages. These icons are cookie-based and will therefore only work when the user visits the OP from a single web browser. Secondly additional client-side software (notably add-ons for web browsers) can be installed to assist the user in noticing potential phishing attacks. This includes tools for helping users to check certificates and domain addresses (such as the address bar highlighting, default in several browsers[5]) and OpenID specific add-ons, such as VeriSign's SeatBelt[6], Sxipper[7] and Microsoft's Identity Selector[8]. A special and creative solution is BeamAuth, suggested by Adida et al. [1] making use of bookmarks and not requiring the user to install additional software.

3.3 Preventing cross-site scripting attacks

Several measures can be taken by RPs and OPs to counter cross-site scripting attacks. Using JavaScript to ensure the RP or OP is not loaded in a (hidden) frame will prevent several of them. Another option for both RPs and OPs is to

[5] This is a default feature of both Internet Explorer and Chrome, and can be installed to Mozilla Firefox as the Locationbar[2] add-on.
[6] `https://pip.verisignlabs.com/seatbelt.do`
[7] `http://www.sxipper.com`
[8] `http://self-issued.info/?p=235`

require a *reduced sign on*, i.e. session-information is not used but the user has to (re-)authenticate himself to the OP for every (or: for this specific) RP.

Since cross-site scripting is no new threat, existing solutions can be used as well. It is for instance recommended to add *nonces* to every user-interaction on the application of the RP. This will prevent an attacker from performing automatic actions with the identity of another user.

3.4 Specification adaptations

A number of problems can be solved or mitigated by addressing them in the OpenID 2.0 specification [19]. The matters of abusing a RP as a port-scanner or performing a DoS on them by presenting a maliciously chosen Identifier URL (Section 2.2) could be addressed by requiring some additional standardisation of the Identifier URL. This could include the refusal of URLs containing port-numbers and local ip-addresses.

A second requirement could be the RP having to set a nonce in the User-Agent initiating the protocol to check whether this nonce still exists at the end of the protocol session, in that way recognizing a Session Swapping attempt (Section 2.1).

A last way of addressing some of the issues is by enforcing secure choices over convenient choices in the OpenID specification. This can be done by changing occurrences of the key words MAY and SHOULD by MUST (as specified in [3]). Examples of this solution are: enforcing the use of HTTPS for communication between OP and RP, and enforcing the use of fragments to prevent identity recycling.

4 Conclusions and future research

In this paper a set of security issues concerning OpenID are discussed, as well as possible solutions to these issues. Applying the low-cost solutions already solves about a third of the issues, making the application of these solutions a strong recommendation to every party using OpenID.

Taking the security issues into account that remain after the application of these low-cost solutions, one can conclude that OpenID was intended and should only be used in applications where a LOA [18] no higher than 1 is required. This level implies that the specification is only to be used in situations where no negative consequences result from erroneous authentication.

Future research could include an investigation to the elements of the OpenID specification that need to be altered in order to make it applicable for applications with a LOA greater than 1 as well. In the context of future internet, the specification would also benefit from becoming even more dynamic and allow for automated negotiations between applications with different LOA levels. Available information suggests that this is most easily achieved when a trust framework is added. Figure 2 shows that when both the whitelisting and the

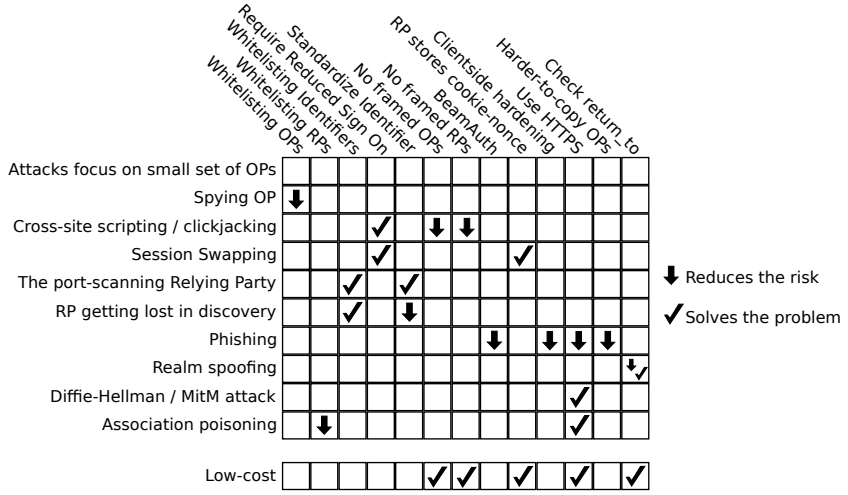

Fig. 2. An overview on which solutions are solving what problems. Low-cost refers to the costs of applying this solution in terms of money, user experience and restriction it places on RPs and OPs (i.e. whitelisting)

low-cost solutions are applied almost all security issues are solved, except for the possibly inescapable phishing problem and value raise of credentials at OPs.

Adding a framework of trust to OpenID lifts it to a different level and solves many of the issues. Doing so moves it into the direction of competing standards such as SAML, WS-Trust, and Information Card. However, such a move likely changes the standard in character, and is a move away from what it was initially intended for: Web SSO where anybody can select their OP and set of RPs of choice. The challenge thus lies in finding a trust model that makes OpenID secure, yet retains OpenID's open and dynamic character.

Acknowledgments: The research described in this paper was carried out in the context of the *cidSafe* project (`http://cidsafe.novay.nl`) which is part of the *Service Innovation & ICT* programme (`http://www.si.i.nl`).

References

1. Ben Adida. BeamAuth: Two-Factor Web Authentication with a Bookmark. In *ACM Conference on Computer and Communications Security*, pages 48–57, 2007.
2. Andrew Arnott. OpenID Association Spoofing. blog.nerbank.net, March 2009. `http://blog.nerdbank.net/2009/03/openid-association-poisoning.html`.
3. S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. RFC 2119, 1997.
4. W. Diffie and M. E. Hellman. New directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22:644–654, 1976.

5. Larry Drebes. Relying Party Stats as of Jan 1st, 2009. JanRain Blog, January 2009. http://blog.janrain.com/2009/01/relying-party-stats-as-of-jan-1st-2008.html.
6. R. Hansen and J. Grossman. Clickjacking. http://www.sectheory.com/clickjacking.htm, September 2008.
7. John Hughes, Scott Cantor, Jeff Hodges, Frederick Hirsch, Prateek Mishra, Rob Philpott, and Eve Maler. Profiles for the OASIS Securit Assertion Markup Language (SAML) V2.0. http://docs.oasis-open.org/security/saml/v2.0/, March 2005.
8. Hyun-Kyung-Oh and Seung-Hun-Jin. The security limitations of SSO in OpenID. In *2008 10th International Conference on Advanced Communication Technology, Gangwon-Do, South Korea, 17-20 Feb. 2008*, pages 1608–1611, Piscataway, NJ, USA, 2008. IEEE.
9. Ashish Jain and Jeff Hodges. Openid review. https://sites.google.com/site/openidreview/, November 2009.
10. Ashish Jain, Andrew Nash, and Jeff Hodges. OpenID Security Issues. Presentation PayPal Information Risk Management, November 2009.
11. Kevin Lawrence and Chris Kaer. WS-Trust 1.4 OASIS Editor Draft. http://docs.oasis-open.org/ws-sx/ws-trust/200802/ws-trust-1.4-ed-01.html, February 2008.
12. Alexander Lindholm. Security Evaluation of the OpenID Protocol. Master's thesis, Sveriges Största Tekniska Universitet, Sweden, 2009.
13. Terry McBride, Dave Silver, Matt Tebo, Chris Louden, and John Bradley. Federal Identity, Credentialing, and Access Management - OpenID 2.0 Profile. Release Candidate 1.0.1, November 2009. http://www.idmanagement.gov/documents/ICAM_OpenID20Profile.pdf.
14. Chris Messina. OpenID Phising Brainstorm. http://wiki.openid.net, December 2008. http://wiki.openid.net/OpenID_Phishing_Brainstorm.
15. Anonymous (most likely Ashish Jain and Jeff Hodges), 2009. Available from https://sites.google.com/site/openidreview/.
16. A. Nanda. Identity Selector Interoperability Profile V1.0, 2007.
17. oauth.net. OAuth Core 1.0 Revision A. http://oauth.net/core/1.0a/, June 2009.
18. Office of Management and Budget (OMB). E-Authentication Guidance for Federal Agencies, Memorandum M-04-04. http://www.whitehouse.gov/omb/memoranda/fy04/m04-04.pdf, December 2003.
19. openid.net. OpenID Authentication 2.0 - Final. http://openid.net/specs/openid-authentication-2_0.html, December 2007.
20. David Recordon and Drummond Reed. OpenID 2.0: A platform for User-Centric Identity Management. In *Conference on Computer and Communications Security Proceedings of the second ACM workshop on Digital identity management*, pages 11–16, Alexandria, Verginia, USA, 2006.
21. Allen Tom and Andrew Arnott. OpenID Security Best Practices. openid.net, July 2009. http://wiki.openid.net/OpenID-Security-Best-Practices.
22. Eugene Tsyrklevich and Vlad Tsyrklevich. Single Sign-On for the Internet: A Security Story. In *BlackHat USA*, 2007.